The listing of claims will replace all prior versions and listings of claims in the application:

Listing of Claims:

1. (currently amended): In an object oriented computing environment, a method of sub-typing an object **performed on a computer having an embedded array and a cache**, comprising:

(a) loading an input object having an object type;

(b) concurrently searching an the embedded array and a the cache for an object sub-typing data structure corresponding to a requested supertype or the input object type; ~~and~~

associating the object sub-typing data structure to the input object,

if the object sub-typing data structure is not found,

(f) searching an overflow array; and

(g) updating the cache with the object sub-typing data structure when the object sub-typing data structure is included in the overflow array.

2. (original): The method as recited in claim 1 wherein the searching of the embedded array is done without checking its size.

3. (original): The method as recited in claim 1 wherein the searching of the embedded array and cache begins from a location(s) indicated by the requested supertype.

4. (original): The method as recited in claim 1 further comprising:

(d) at compile-time, loading a location description of a known-constant supertype; and

(e) incorporating the result of the location description loading into instructions produced by a compiler.

5. (original): The method as recited in claim 4 wherein the location description is applied at compile time to a known-constant input object type.

6. (original): The method as recited in claim 5 wherein the location description corresponds to an embedded array location.

7. (original): The method as recited in claim 5 wherein the location description corresponds to a cache location.

8. (original): The method as recited in claim 5 wherein the location description corresponds to an overflow array location.

9. (Canceled)

10. (original) The method recited in claim 1, further comprising:

if the requested supertype is a special type,

(h) searching only the cache for the object sub-typing data structure corresponding to the input object type.

11. (original): The method recited in claim 1, further comprising:

if the requested supertype is not a special object type,

(i) searching only the embedded array for the object sub-typing data structure corresponding to the input object type.

12. (currently amended): The method recited in **claim 11** ~~**claim 1**~~, further comprising:

if the input object type is the special object type,

comparing input object type directly against the requested supertype, and returning a match if the comparison is successful.

13. (currently amended): The method recited in **claim 11** ~~claim 1~~, further comprising:

if the input object type is the special object type,

omitting the object sub-typing data structure(s) corresponding to the input object type from an overflow array associated with the input object type.

14. (currently amended): The method recited in **claim 11** ~~claim 1~~, further comprising:

if the input object type is the special object type,

sharing an overflow array between at least two object types when the object type search requires the same overflow array contents.

15. (original): The method recited in claim 1, further comprising:

forcing the object sub-typing data structure to be not-special by assigning it a location in the embedded array.

16. (original): The method recited in claim 10, further comprising:

if the object sub-typing data structure is not found in the search (h),

(j) searching the overflow array for the object sub-typing data structure.

17. (original): The method recited in claim 16, further comprising:

updating the cache based upon the searching (j).

18. (original): The method recited in claim 10, further comprising:

if the object sub-type is found in the searching (h), associating the object sub-typing data structure to the input object.

19. (currently amended): The method recited in **claim 1** ~~**claim 9**~~, wherein the cache is updated by adding the object sub-typing data structure found in the searching (f).

20. (currently amended): The method recited in **claim 1** ~~**claim 9**~~, wherein the cache is updated by deleting an object sub-typing data structure, the cache being further updated by adding the object sub-typing data structure found in the searching (f).

21. (canceled)

22. (currently amended): ~~**The apparatus recited in claim 21, further comprising:**~~**In an object oriented computing environment, an apparatus for sub-typing an object in a computer having an embedded array and a cache, comprising:**

means for receiving an input object having an object type;

means for searching the embedded array and the cache for an object sub-typing data structure corresponding to the input object type;

means for associating the object sub-typing data structure to the input object

if the input object sub-type is not found in the searching ~~**(l)**~~,

~~**(n)**~~ means for searching~~**, wherein**~~ an overflow array ~~**is searched**~~ for an object sub-typing data structure corresponding to the input object type; **and**

~~**(o)**~~ means for updating the cache contents based on object sub-typing data structure~~**(s)**~~ found in the searching ~~**(n)**~~.

23. (currently amended): An object sub-typing system included in a computer **having an embedded array containing a first group of object sub-typing data structures and a cache containing a second group of object sub-typing data structures,** comprising ~~of~~:

~~an input object having an object type;~~

~~an embedded array containing a first group of object sub-typing data structures;~~

~~a cache containing a second group of object sub-typing data structures; and~~

an object locator unit arranged to search the embedded array and cache for **a** ~~the~~ requested **object** supertype~~, the object locator unit further having connections to modify the contents of the cache~~; **and**

a cache modifier unit coupled to the object locator unit arranged to modify the contents of the cache, wherein any sub-typing data structures in the first group of objects in the embedded array that have a depth greater than a fixed depth are added to the cache.

24. (original): The system recited in claim 23, further comprising:

an overflow array containing a third group of object sub-typing data structures, wherein the overflow array is arranged to be searched by the object locator unit.

25. (original): The system recited in claim 23, further comprising:

a program execution unit coupled to the object locator unit, wherein the connection is configured to enable the program execution unit to request object types and receive object-handling information from the object locator unit.

26. (original): The system recited in claim 25, wherein the embedded array is contained within an object type data structure associated with a computer program being executed by the program execution unit.

27. (original): The system recited in claim 23, wherein the first group of object sub-typing data structures and the second group of object sub-typing data structures are mutually exclusive.

28. (original): The system recited in claim 23, wherein the second group of object sub-typing data structures is selected from the third group of object sub-typing data structures.

29. (original): The system recited in claim 23, wherein the cache is initialized to contain only object sub-typing data structures of a special type.

30. (original): The system recited in claim 23, wherein the second group of object sub-typing data structures in the cache are the object sub-typing data structures substantially expected to be searched for by the object locator unit.

31. (original): The system recited in claim 23, wherein the first group of object sub-typing data structures in the embedded array has a hierarchical data structure, wherein the hierarchical data structure has a fixed depth.

32. (canceled)

33. (currently amended): A computer program product for obtaining information associated with a particular object sub-type, the computer program product comprising:

computer code that determines the object type of an input object;

computer code that searches a cache and an embedded array for an object sub-typing data structure corresponding to the input object type;

computer code that retrieves and makes available information associated with the found object sub-typing data structure;

computer code that searches an overflow array if the sub-typing data structure corresponding to the input object type is not found in the embedded array and cache;

computer code that updates the cache contents based on object sub-typing data structure(s) found in the overflow array search; and

a computer-readable medium that stores the computer code.

34. (canceled)

35. (canceled)

36. (original): A computer program product according to claim 33 wherein the computer-readable medium is one selected from the group consisting of a data signal embodied in a carrier wave, a CD-ROM, a hard disk, a floppy disk, a tape drive, and semiconductor memory.

37. (new): In an object oriented computing environment, computer program product for efficiently sub-typing an object executed on a computer having an embedded array and a cache, comprising:

computer code for loading an input object having an object type;

computer code for concurrently searching an the embedded array and a the cache for an object sub-typing data structure corresponding to a requested supertype or the input object type;

computer code for associating the object sub-typing data structure to the input object, if the object sub-typing data structure is not found,

computer code for searching an overflow array;

computer code for updating the cache with the object sub-typing data structure when the object sub-typing data structure is included in the overflow array; and

computer readable medium for storing the computer code.

38. (new): Computer program product as recited in claim 37 wherein the searching of the embedded array is done without checking its size.

39. (new): Computer program product as recited in claim 37 wherein the searching of the embedded array and cache begins from a location indicated by the requested supertype.

40. (new): Computer program product as recited in claim 37 further comprising:

computer code for loading a location description of a known-constant supertype at compile-time; and

computer code for incorporating the result of the location description loading into instructions produced by a compiler.

41. (new): Computer program product as recited in claim 40 wherein the location description is applied at compile time to a known-constant input object type.

42. (new): Computer program product as recited in claim 41 wherein the location description corresponds to an embedded array location.

43. (new): Computer program product as recited in claim 42 wherein the location description corresponds to a cache location.

44. (new): Computer program product as recited in claim 42 wherein the location description corresponds to an overflow array location.

45. (new): Computer program product recited in claim 37, further comprising:

computer code for searching only the cache for the object sub-typing data structure corresponding to the input object type if the requested supertype is a special type.

46. (new): Computer program product recited in claim 37, further comprising:

computer code for searching only the embedded array for the object sub-typing data structure corresponding to the input object type if the requested supertype is not a special object type.

47. (new): Computer program product recited in claim 46, further comprising:

computer code for comparing input object type directly against the requested supertype, and returning a match if the comparison is successful if the input object type is the special object type.

48. (new): Computer program product recited in claim 46, further comprising:

computer code for omitting the object sub-typing data structure(s) corresponding to the input object type from an overflow array associated with the input object type if the input object type is the special object type.

49. (new): Computer program product recited in claim 46, further comprising:

computer code for sharing an overflow array between at least two object types when the object type search requires the same overflow array contents if the input object type is the special object type,.

50. (new): Computer program product recited in claim 37, further comprising:

computer code for forcing the object sub-typing data structure to be not-special by assigning it a location in the embedded array.

51. (new): Computer program product recited in claim 47, further comprising:

computer code for searching the overflow array for the object sub-typing data structure if the object sub-typing data structure is not found in the searching only the embedded array for the object sub-typing data structure corresponding to the input object type if the requested supertype is not a special object type.

52. (new): Computer program product recited in claim 51, further comprising:

updating the cache based upon the searching the overflow array for the object sub-typing data structure.

53. (new): Computer program product recited in claim 10, further comprising:

computer code for associating the object sub-typing data structure to the input object if the object sub-type is found in the searching only the cache for the object sub-typing data structure corresponding to the input object type.

54. (new): Computer program product recited in claim 37 wherein the cache is updated by adding the object sub-typing data structure found in the searching of the overflow array.

55. (new): Computer program product recited in claim 37, wherein the cache is updated by deleting an object sub-typing data structure, the cache being further updated by adding the object sub-typing data structure found in the searching of the overflow array.

56. (new): In an object oriented computing environment, a method performed on a computer having an embedded array and a cache of sub-typing an object, comprising:

loading an input object having an object type;

concurrently searching an the embedded array and a the cache for an object sub-typing data structure corresponding to a requested supertype or the input object type;

associating the object sub-typing data structure to the input object; and

forcing the object sub-typing data structure to be not-special by assigning it a location in the embedded array.